Unsupervised Learning for customer segmentation: comparing advanced clustering methods in practice

What changes when K-Means, K-Medoids, Ward hierarchical clustering, DBSCAN and Gaussian Mixtures see the same 8,950 credit-card customers?

Dataset source: Arjun Bhasin — Credit Card Dataset for Clustering, Kaggle (CC0: Public Domain).

The Kaggle data card describes roughly 9,000 active credit-card holders observed over the previous six months, with 18 customer-level behavioral variables. The local file used in this exercise contains 8,950 rows and the same 18 columns.

The question behind the exercise

If I keep the data, preprocessing and dimensionality-reduction space fixed, how much of the final segmentation is determined by the clustering algorithm itself?

Most introductory clustering examples end with one model and one colorful scatter plot. I find this useful for learning the API, but it tends to hide the most difficult part of unsupervised learning: there is no target label telling us that one partition is “correct.” An algorithm is therefore not simply a tool that discovers a segmentation already hidden in the data. The nature of the algorithm itself defines what a cluster should look like.

For this exercise, I keep the dataset and modeling space fixed and change only the clustering logic. K-Means looks for compact groups around centroids. K-Medoids replaces those synthetic centers with real observations. Ward hierarchical clustering builds a nested tree through successive merges. DBSCAN defines groups through local density and can reject observations as noise. Gaussian Mixture Models treat membership probabilistically and allow each component to have its own covariance geometry.

For this reason, the goal is comparative. The point is not to establish which algorithm “wins”, but to understand what each algorithm sees in the same space, why it produces that specific partition and, above all, which evidence allows us to defend the final choice.

1. First, make the comparison fair: one common modeling space

The raw dataset contains 17 behavioral variables after removing CUST_ID from the feature matrix. Monetary and transaction variables are strongly right-skewed, so extreme but still valid observations can dominate Euclidean distances. Instead of automatically removing every IQR outlier, I preferred to preserve high-activity customers and reduce their leverage with log1p on selected non-negative variables.

```
log_transform_cols = [
    "BALANCE", "PURCHASES", "ONEOFF_PURCHASES",
    "INSTALLMENTS_PURCHASES", "CASH_ADVANCE",
    "CASH_ADVANCE_TRX", "PURCHASES_TRX",
    "CREDIT_LIMIT", "PAYMENTS", "MINIMUM_PAYMENTS"
```

```
]

for col in log_transform_cols:
    X_transformed[col] = np.log1p(X_transformed[col].clip(lower=0))

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_transformed)
```

Technical note — *log1p compresses the right tail while preserving order. StandardScaler then puts the features on a comparable mean-zero, unit-variance scale. For distance-based clustering this is not cosmetic, because scale directly defines the geometry.*

The next step is PCA. In the notebook I use PCA as the modeling space, not only as a plotting trick. The fitted transformation retains eight principal components to reach 90% explained variance; the actual retained variance is 91.48%. In this way I reduce correlated information while preserving most of the transformed variance.

```
pca = PCA(n_components=0.90, random_state=RANDOM_STATE)
X_pca = pca.fit_transform(X_scaled)

print(X_pca.shape)
# (8950, 8)
```

Technical note — *PCA creates orthogonal components. I later use only two dimensions for visual inspection, but all models are fitted on the eight retained components.*

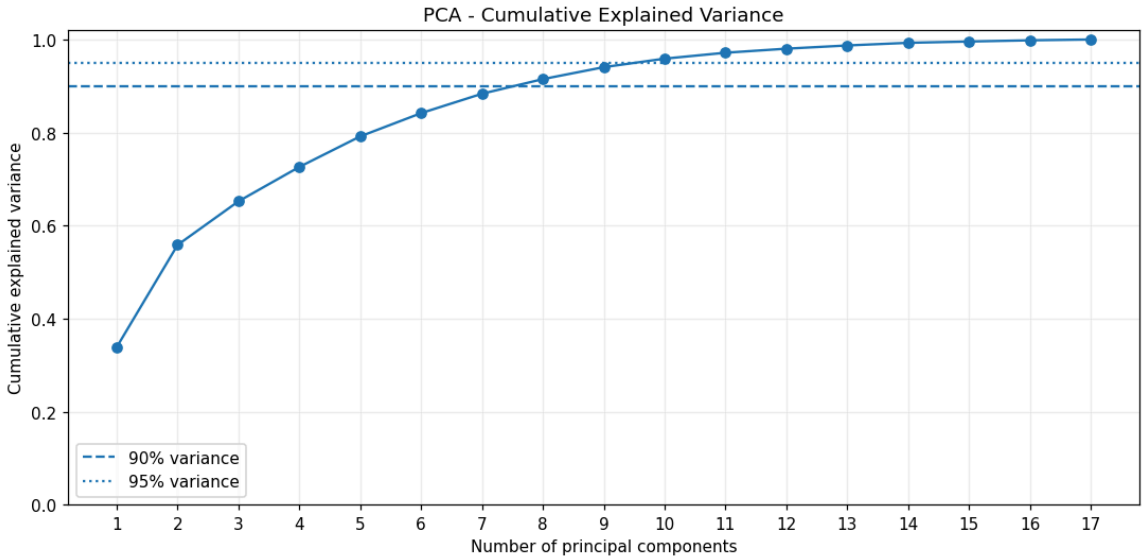


Figure 1 — PCA cumulative explained variance. Eight components cross the 90% threshold, while ten are required for 95%. The clustering comparison below uses the eight-dimensional PCA representation.

2. Why K = 5? Because “best K” depends on what you mean by best

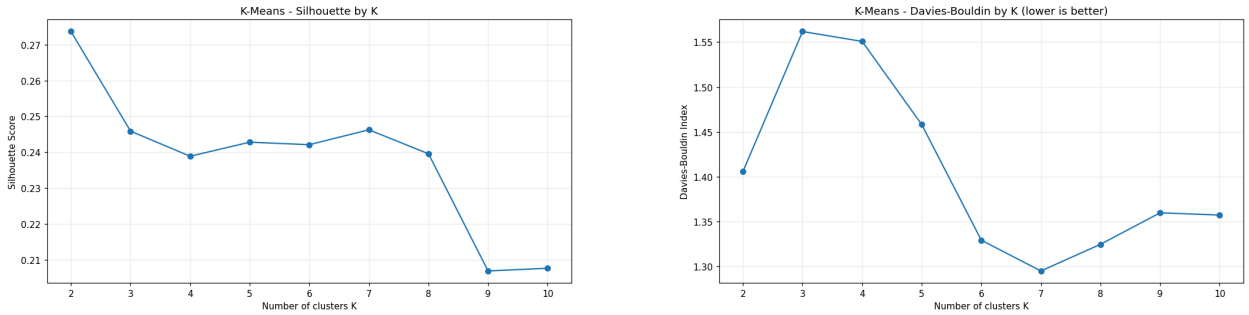
I use K-Means as the baseline because it gives a simple and stable reference partition. I tested K from 2 to 10 and recorded inertia, Silhouette Score, Davies-Bouldin Index (DBI), and Calinski-Harabasz (CH). The

metrics, however, do not point to a single optimum. K=2 has the highest Silhouette Score (0.274). DBI reaches its lowest value at K=7 (1.295). K=5 sits between these two extremes, with Silhouette 0.243, DBI 1.458 and CH 2,514.014.

```
for k in range(2, 11):
    model = KMeans(n_clusters=k, n_init=30, random_state=RANDOM_STATE)
    labels = model.fit_predict(X_pca)

    kmeans_results.append({
        "K": k,
        "Inertia": model.inertia_,
        "Silhouette": silhouette_score(X_pca, labels),
        "Davies-Bouldin": davies_bouldin_score(X_pca, labels),
        "Calinski-Harabasz": calinski_harabasz_score(X_pca, labels)
    })
```

Technical note — *Silhouette rewards cohesion and separation; lower DBI indicates more compact and separated clusters; CH compares between-cluster dispersion with within-cluster dispersion. Inertia is useful for reading the elbow, but it cannot be compared across unrelated models.*


K-Means - Silhouette by K
Silhouette Score
Number of clusters K
K-Means - Davies-Bouldin by K (lower is better)
Davies-Bouldin Index
Number of clusters K

Silhouette by K **Davies-Bouldin by K**

Figure 2 — K selection cannot be reduced to one metric. The coarser K=2 partition is cleaner by Silhouette, while DBI favors a more granular K=7. I retain K=5 as a controlled five-segment reference for the cross-model comparison.

The stronger argument for K=5 is its stability. The final K-Means model uses n_init=50, then I re-fit the five-cluster solution across ten random seeds and compare each partition with the reference using Adjusted Rand Index (ARI). The mean ARI is 0.9957 and the minimum is 0.9922. This means that, on this PCA representation, the five-way partition is extremely insensitive to initialization.

```
reference_labels = labels_kmeans

for seed in range(10):
    labels_seed = KMeans(
        n_clusters=5, n_init=10, random_state=seed
    ).fit_predict(X_pca)

    score = adjusted_rand_score(reference_labels, labels_seed)
```

Technical note — *ARI compares two partitions while correcting for the agreement expected by chance. A value near 1 means almost identical assignments, even when numeric cluster labels are permuted.*

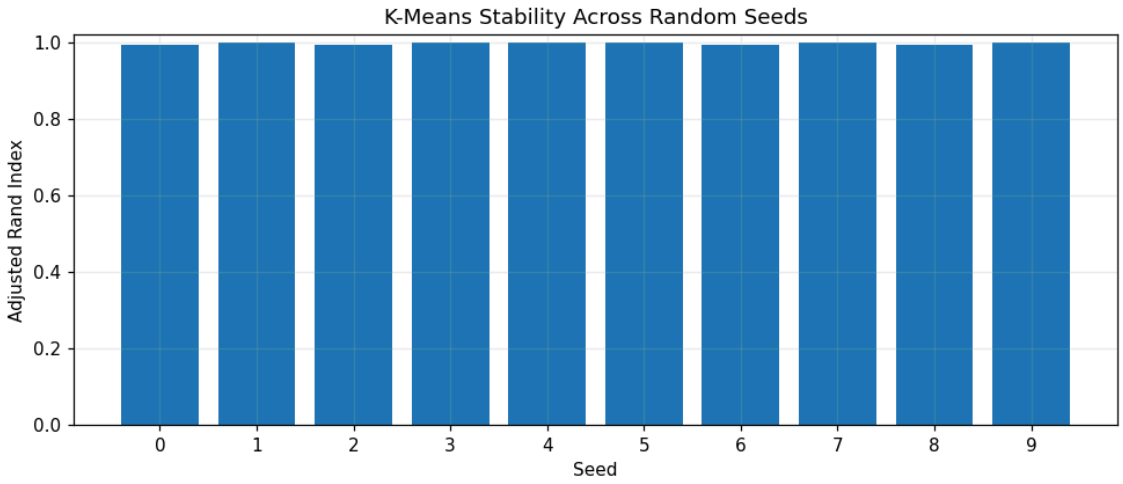


Figure 3 — K-Means stability across random seeds. The partition remains almost unchanged, making K=5 a useful reference for comparing algorithms rather than simply comparing random initializations.

3. Five algorithms, five definitions of a cluster

At this point the comparison becomes more interesting: I keep X_pca fixed and change only the clustering rule. The five PCA projections below are not simply five different visual styles applied to the same segmentation. They are five different partitions produced by five different assumptions.

K-Means — centroid geometry

K-Means assigns every point to one of five centroids and minimizes squared Euclidean distance within clusters. With K=5 it produces the strongest overall internal metrics in the final comparison: Silhouette 0.243, DBI 1.458 and CH 2,514.014. Cluster shares range from 14.57% to 27.16%, so no segment collapses into a very small residual group.

```
kmeans_final = KMeans(
    n_clusters=5,
    n_init=50,
    random_state=RANDOM_STATE
)
labels_kmeans = kmeans_final.fit_predict(X_pca)
```

Technical note — *n_init=50 runs K-Means from multiple initial centroid configurations and keeps the solution with the lowest inertia. In this way I reduce sensitivity to a poor random start.*

K-Medoids — real observations as centers

In the notebook, I implemented a CLARA-style K-Medoids approximation because exact full-data PAM is expensive at this scale. The method samples 1,200 observations three times, initializes candidate medoids near the K-Means centers, iteratively replaces each medoid with the observation minimizing within-cluster

pairwise distance, and finally assigns the full dataset to the best medoid set identified. The result is very close to K-Means: Silhouette 0.239 and DBI 1.508.

```
d_to_medoids = pairwise_distances(X_sample, X_sample[medoid_idx])
labels_sample = d_to_medoids.argmin(axis=1)

cluster_points = X_sample[members]
pairwise = pairwise_distances(cluster_points)
best_local = members[np.argmin(pairwise.sum(axis=1))]
```

Technical note — *A medoid is an actual observation, not the arithmetic mean. This makes the center less vulnerable to extreme values, although CLARA trades exactness for scalability through sampling.*

Ward hierarchical clustering — build the hierarchy, then cut it

Hierarchical clustering is conceptually different from K-Means because it does not need K to build the tree. Ward linkage repeatedly merges the pair of groups that causes the smallest increase in within-cluster variance. In the notebook I generate a dendrogram from a 700-customer sample for visual inspection, then AgglomerativeClustering cuts the full-data hierarchy into five clusters for a controlled comparison. Its Silhouette falls to 0.176 and DBI rises to 1.769.

```
Z = linkage(X_pca[dendro_idx], method="ward")

hierarchical = AgglomerativeClustering(
    n_clusters=5,
    linkage="ward"
)
labels_hierarchical = hierarchical.fit_predict(X_pca)
```

Technical note — *Ward linkage minimizes variance and therefore still favors compact Euclidean groups. The greedy merge history, however, can produce a different final partition from centroid optimization.*

DBSCAN — density replaces K

DBSCAN removes the need to specify a fixed number of clusters. Instead, local density is defined by two parameters: eps, the neighborhood radius, and min_samples, the minimum number of observations required to form a dense region. Here min_samples is set to twice the PCA dimensionality (16), while eps is estimated from the knee of the 16-nearest-neighbor distance curve: 1.7504.

```
MIN_SAMPLES = max(10, 2 * X_pca.shape[1])

neighbors = NearestNeighbors(n_neighbors=MIN_SAMPLES)
neighbors.fit(X_pca)
distances, _ = neighbors.kneighbors(X_pca)
k_distances = np.sort(distances[:, -1])

EPS_KNEE = float(k_distances[knee_index])

dbscan_final = DBSCAN(
```

```
    eps=EPS_KNEE,
    min_samples=MIN_SAMPLES,
    n_jobs=-1
)
```

Technical note — *DBSCAN discovers three non-noise clusters and labels 5.66% of customers as noise. The largest cluster contains 93.30% of all observations. This clearly shows that the density structure in this PCA space is very different from the five-group centroid view.*

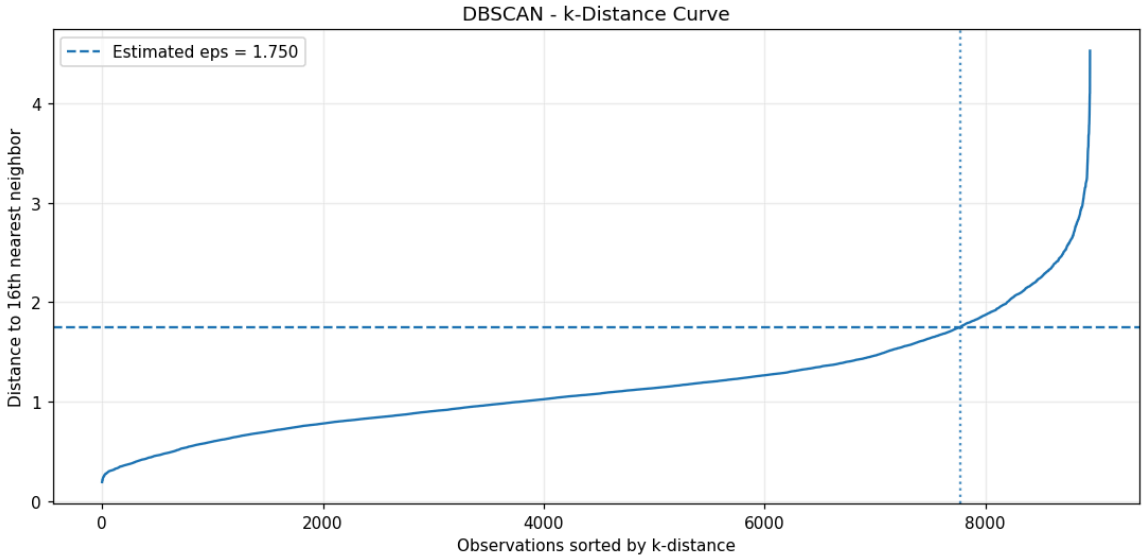


Figure 4 — DBSCAN k-distance curve. The estimated knee at eps ≈ 1.750 provides a data-driven neighborhood radius; changing eps changes both the number of dense groups and the share of noise.

Gaussian Mixture Model — probability instead of only a hard label

GMM starts from a different assumption: the data are generated by a mixture of Gaussian components. Unlike K-Means, each component can have its own full covariance matrix, so groups can be elliptical and oriented differently. In the notebook I evaluate 2–8 components with AIC and BIC, but retain five components for the controlled comparison. This distinction is important: five is the common experimental resolution, not the AIC/BIC optimum.

```
gmm_final = GaussianMixture(
    n_components=5,
    covariance_type="full",
    n_init=5,
    max_iter=500,
    reg_covar=1e-6,
    random_state=RANDOM_STATE
)

labels_gmm = gmm_final.fit_predict(X_pca)
gmm_probabilities = gmm_final.predict_proba(X_pca)
```

Technical note — *predict_proba preserves membership uncertainty. The mean maximum membership probability is 0.97, while only 1.41% of observations have maximum membership probability below 0.60.*

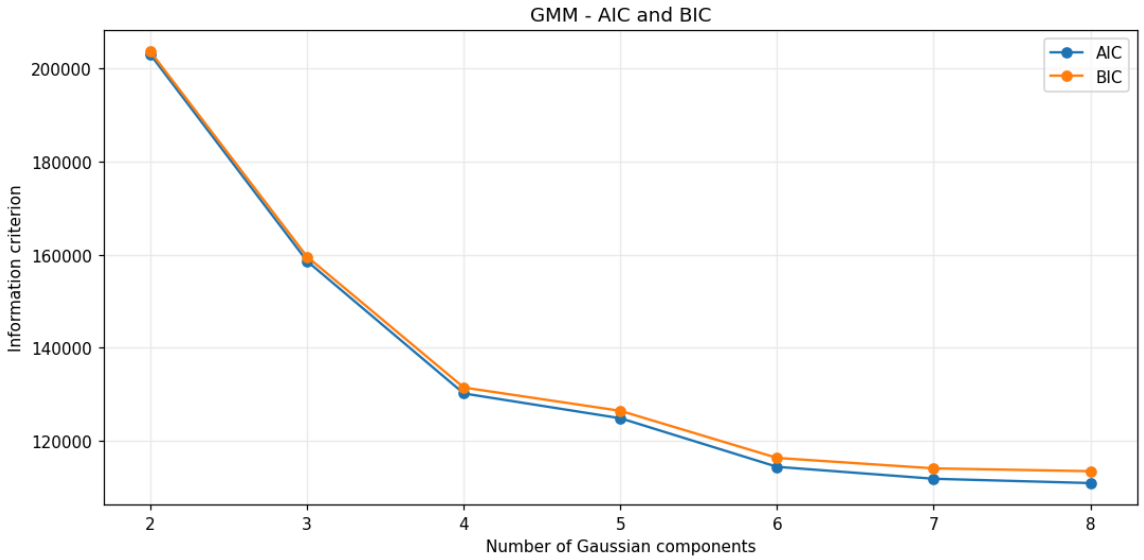
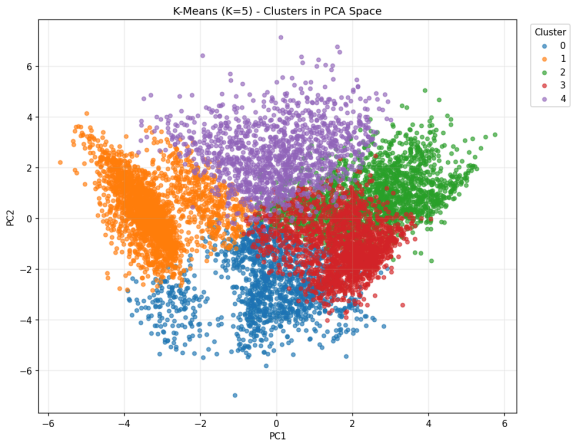


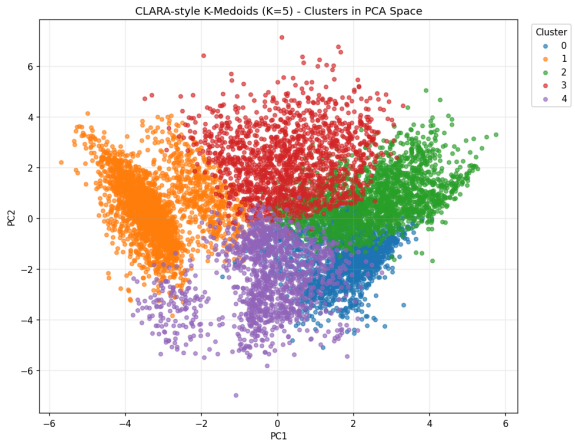
Figure 5 — GMM AIC and BIC across 2–8 components. Both criteria continue to decline through the tested range, so the five-component model should be interpreted as a controlled comparison choice, not as a unique information-criterion optimum.

The same PCA space now tells five different stories

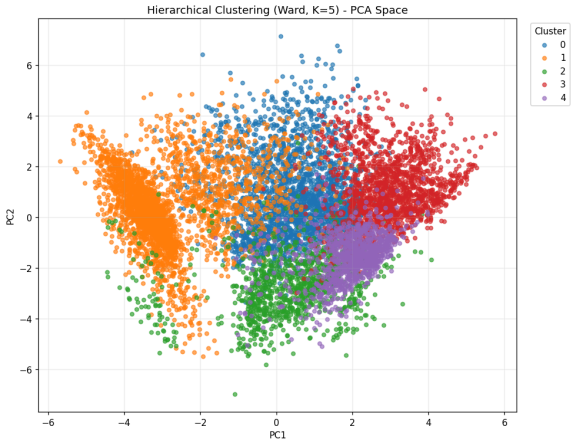
When the five PCA projections are placed together, the geometry becomes much easier to read. K-Means and K-Medoids create visibly similar partitions. Ward hierarchical clustering preserves the broad structure but changes several boundaries. GMM redistributes membership through covariance-aware components. DBSCAN, instead, effectively says that most customers belong to one connected dense region, with only small secondary groups and noise.



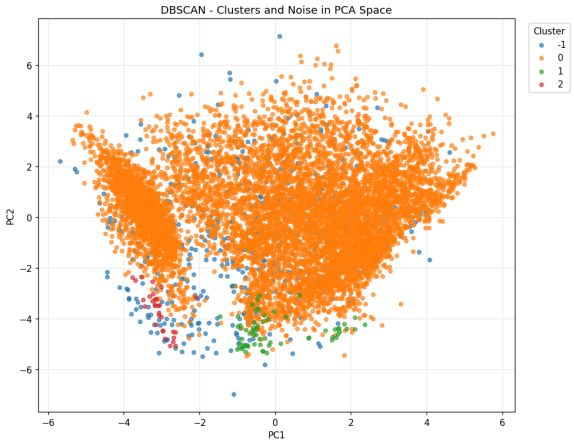
K-Means, K=5



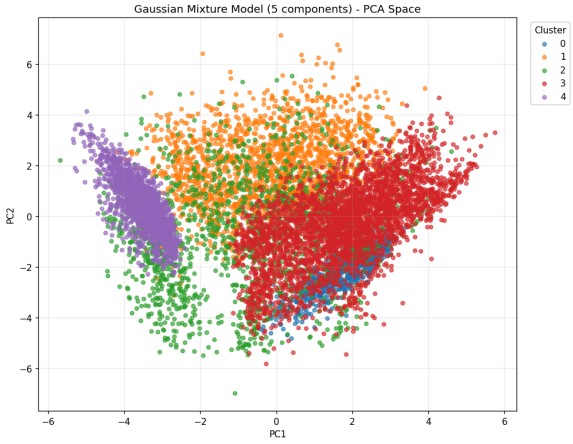
CLARA-style K-Medoids, K=5



Ward hierarchical, K=5



DBSCAN, tuned eps



GMM, 5 components

Figure 6 — Five clustering algorithms on the same eight-dimensional PCA representation, shown through PC1 and PC2. Cluster numbers are local labels, so they must not be compared by color or numeric ID across models.

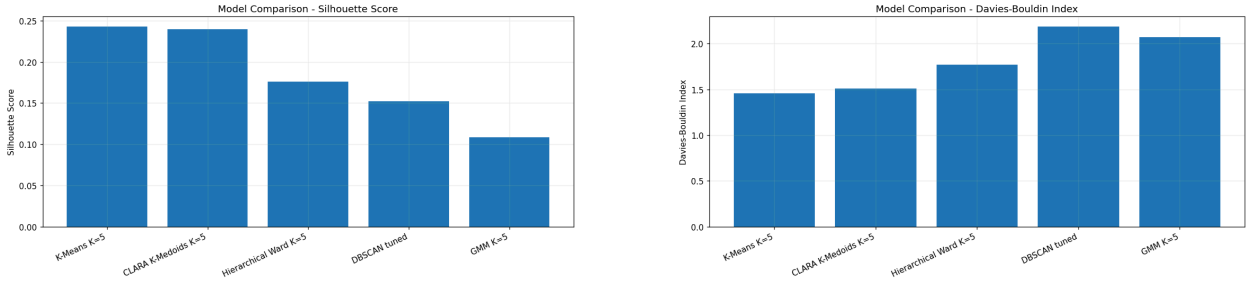
4. Internal metrics are only half the comparison

Internal metrics answer one useful question: how compact and separated is each partition in the modeling space? But they do not answer another one, equally important: did two algorithms actually assign the same customers together?

Model	Clusters	Sil.	DBI	CH	Noise	Smallest	Largest
K-Means K=5	5	0.243	1.458	2,514.014	0.00%	14.57%	27.16%
CLARA K-Medoids K=5	5	0.239	1.508	2,450.578	0.00%	14.38%	27.90%
Hierarchical Ward K=5	5	0.176	1.769	1,851.790	0.00%	12.18%	31.72%
DBSCAN tuned	3	0.152	2.187	148.189	5.66%	0.34%	93.30%

GMM K=5	5	0.109	2.071	1,349.347	0.00%	8.59%	37.21%

Table 1 — Final cross-model comparison from the notebook. Higher Silhouette and CH are better, while lower DBI is better. DBSCAN is evaluated here with noise treated as a label so that the table reflects the complete partition.



Silhouette Score **Davies-Bouldin Index**

Figure 7 — According to the internal metrics, K-Means and K-Medoids form the strongest pair. Hierarchical is intermediate, while DBSCAN and GMM are weaker under this specific common representation and evaluation setup.

The ARI matrix, however, changes the discussion. K-Means and K-Medoids have ARI 0.805: replacing centroids with medoids changes some boundaries, but the dominant structure remains similar. Hierarchical has moderate agreement with K-Means (0.457) and K-Medoids (0.477). GMM is further away: ARI 0.319 with K-Means and 0.348 with K-Medoids. DBSCAN is almost orthogonal to the centroid/medoid view, with ARI around 0.012.

```
score = adjusted_rand_score(
    agreement_labels[name_i],
    agreement_labels[name_j]
)
```

Technical note — *ARI is label-invariant: cluster "0" in one model does not need to correspond to cluster "0" in another. It measures agreement in pairwise assignments rather than matching numeric labels.*

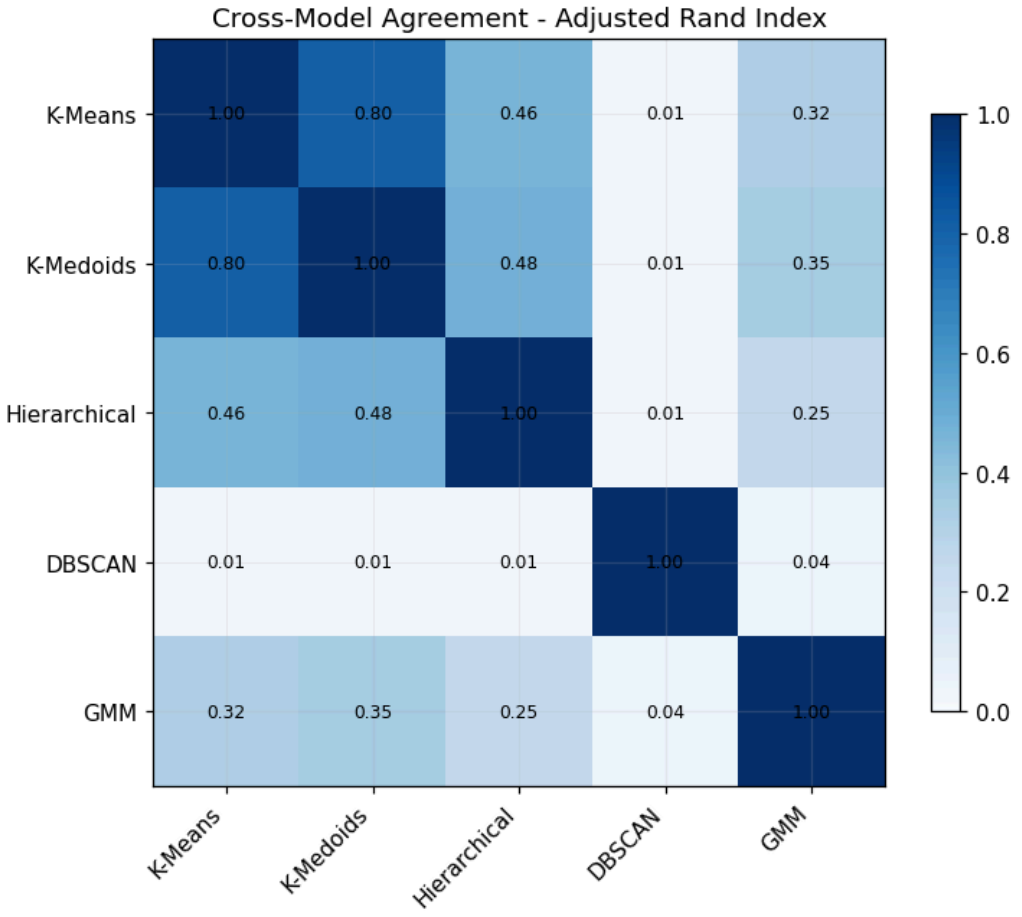


Figure 8 — Cross-model Adjusted Rand Index. The strongest agreement is K-Means vs K-Medoids (0.805). DBSCAN has near-zero agreement with the other main partitions, confirming that its density assumption identifies a fundamentally different structure.

The key result

"Five clusters" are not a universal object hidden inside the dataset. They are a model-dependent partition. For this reason, the strongest comparison combines internal quality, stability, cluster balance and direct agreement between assignments.

5. Why K-Means becomes the final descriptive lens

Selecting K-Means does not mean that the other models are "wrong." The evidence supports a more specific conclusion: on this transformed eight-dimensional PCA representation, and for a five-segment descriptive objective, K-Means provides the best combination of internal metrics, stability, balance and interpretability. K-Medoids represents a very strong robustness check because it reaches an almost identical structure using real observations as centers. Hierarchical adds a useful nested view. DBSCAN, instead, demonstrates that the data do not naturally decompose into five density islands. GMM finally adds information about uncertainty at cluster boundaries.

The final step consists of moving back from PCA space to the original variables. This is where abstract clusters become interpretable behavioral profiles. Cluster 2 has the highest average PURCHASES (2,788.82), 35.68 purchase transactions and a CREDIT_LIMIT of 6,252.88, while average CASH_ADVANCE is only 47.20. Cluster 1 shows the opposite pattern: average PURCHASES of only 41.05, but CASH_ADVANCE of 1,988.16. Cluster 3 is characterized by frequent installment purchases, with PURCHASES_INSTALLMENTS_FREQUENCY of 0.77. Cluster 4, finally, combines the highest BALANCE (3,404.96) with the highest CASH_ADVANCE (2,735.49).

Cluster	Descriptive label	Customers	Share	Selected mean signals
0	Lower-activity / lower-balance	1,466	16.38%	BALANCE 187.63; PURCHASES 326.38
1	Cash-advance-intensive	2,431	27.16%	CASH_ADVANCE 1,988.16; PURCHASES 41.05
2	High purchase-amount	1,906	21.30%	PURCHASES 2,788.82; PURCHASES_TRX 35.68
3	Frequent installment-purchase	1,843	20.59%	INSTALLMENT_FREQ 0.77; PURCHASES 649.61
4	Higher-balance mixed-activity	1,304	14.57%	BALANCE 3,404.96; CASH_ADVANCE 2,735.49

Table 2 — Final K-Means descriptive segments. The labels come from observed behavior; they are not credit-risk, profitability or solvency diagnoses.

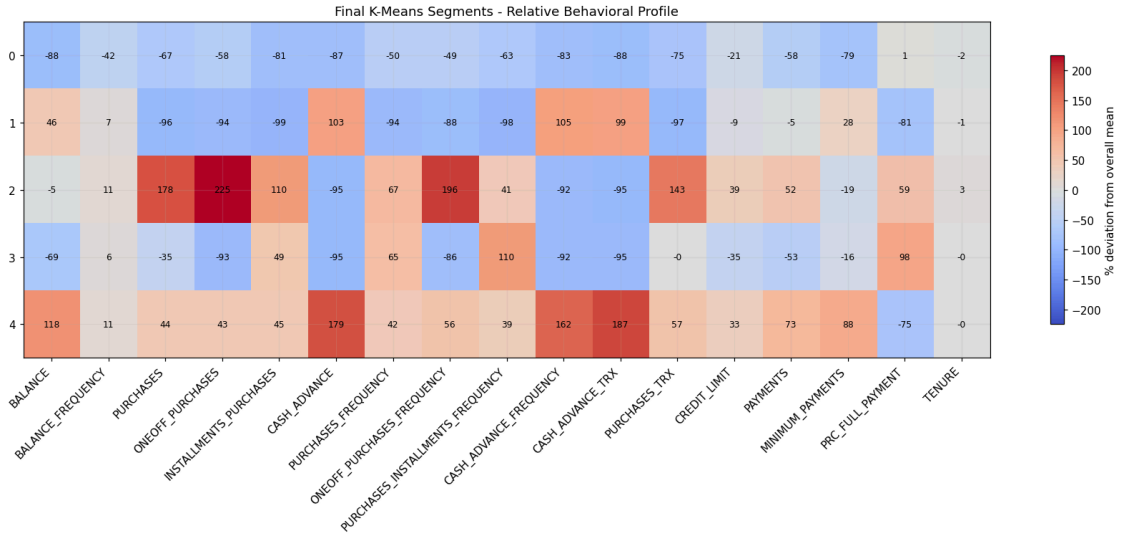


Figure 9 — Final K-Means relative behavioral profile. Each cell shows the percentage deviation of the cluster mean from the overall mean. The heatmap makes the five behavioral signatures easier to interpret than the raw PCA coordinates.

I use t-SNE only for visualization, as another nonlinear view of the final K-Means labels. It is useful as a diagnostic picture of local neighborhoods, but I deliberately do not use it as the clustering space because its geometry depends strongly on hyperparameters such as perplexity and because global distances in t-SNE are not directly interpretable like PCA coordinates.

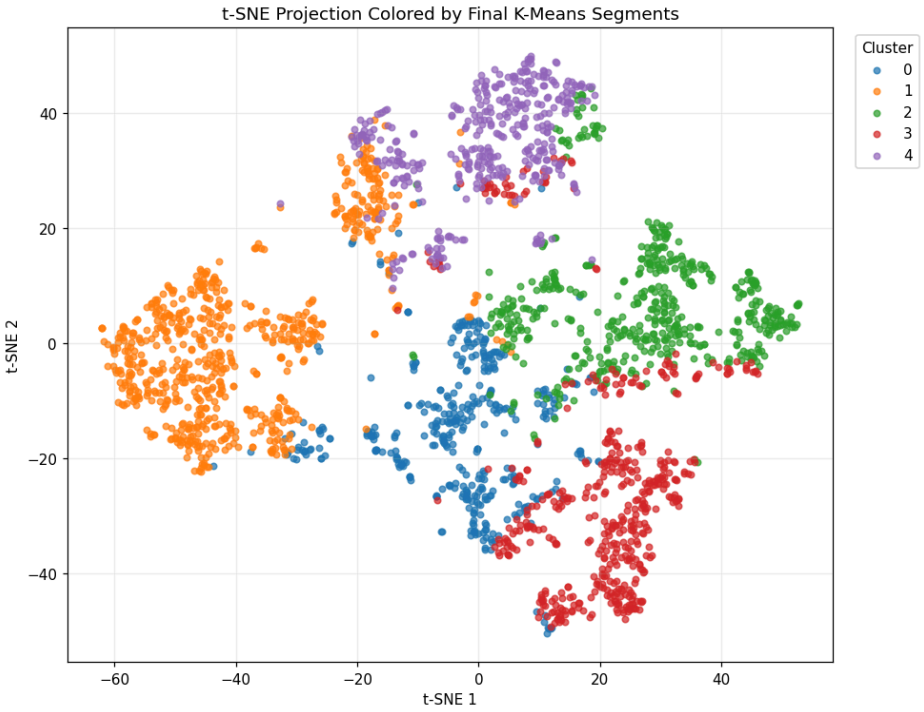


Figure 10 — t-SNE projection colored by the final K-Means segments. The chart is useful for visually inspecting local separation, but all clustering decisions in this exercise were made in the eight-dimensional PCA space.

What this exercise changed in the way I think about unsupervised learning

- **Preprocessing is part of the model.** Log transforms, scaling and dimensionality reduction change the distances before a clustering algorithm sees the data.
- **Do not choose K from one chart.** Inertia, Silhouette, DBI, CH, balance and stability answer different questions and should be read together.
- **Compare partitions, not only scores.** ARI shows that K-Means and K-Medoids largely agree, while DBSCAN is describing something structurally different.
- **Density methods need density-aware interpretation.** A good non-noise DBI does not automatically compensate for one cluster absorbing more than 90% of the sample.
- **Soft clustering contains additional information.** GMM membership probabilities preserve uncertainty that a hard label discards.
- **Interpret the result in the original variables.** PCA is the modeling geometry; BALANCE, PURCHASES, CASH_ADVANCE and frequencies are the language in which the segments can actually be explained.

The broader lesson, for me, is that unsupervised learning rarely provides a single certificate of truth. Confidence comes from triangulating multiple sources of evidence: sensitivity to preprocessing, internal metrics, stability across random initialization, cluster balance, agreement across algorithms, and interpretability in the original feature space. In this exercise, K-Means is useful not because it “found the true five clusters”, but because several independent checks make its five-cluster partition the most defensible descriptive lens among the tested alternatives.